Tiva Based Daughter Board for Firebird V Software Manual

ERTS Lab IIT Bombay

July 7, 2017

1 Credits

Version 1.0
July 7, 2017

Documentation Author(Alphabetical Order):

1. Ayush Gaurav, Intern eYSIP 2017
2. Nagesh K, Intern eYSIP 2017

Credits(Alphabetical Order):

1. Prof Kavi Arya, CSE IIT Bombay
2. Nex Robotics Pvt. Ltd.
3. Piyush Manavar, Team e-Yantra
4. Saurav Shandilya, Team e-Yantra

2 Notice

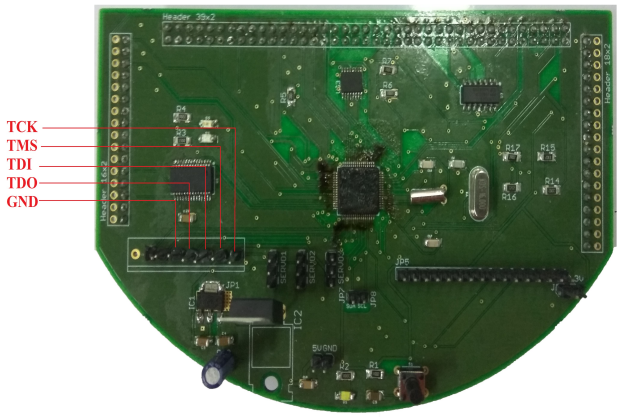
The contents of this manual are subject to change without notice. All efforts have been made to ensure the accuracy of contents in this manual. However, should any errors be detected, e-yantra welcomes your corrections. You can send us your queries / suggestions at Contact Us

Table Of Content

3 Software Manual

3.1 Programming the bot

The uC based board does not have an onboard programmer. So the JTAG pins tapped from a functional TIVA launchpad is connected to the JTAG headers present on the Daughter board. These connections are shown in the Figure1 and Figure2



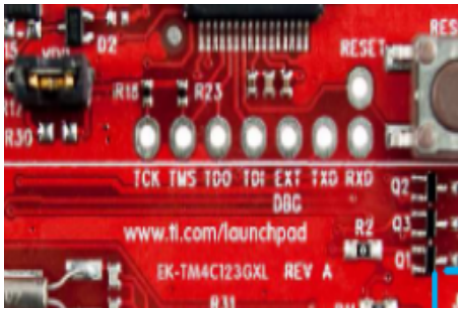


Figure1 Figure2

The TCK, TMS, TDI and TDO pins of the Daughter Board should be connected to the corresponding TCK, TMS, TDI and TDO pins of the TIVA launchpad.

3.2 Simple I/O Operation

In this section, we will learn how to initialize GPIO pins in TM4C123G and use them for simple input-output operations. The code for TIVA is written in C99 programming standards with Hungarian standard for naming variables.

3.2.1 Header Files

Before writing the code we have to include certain header files to access the TivaWare API's. These header files are required for variable definition. The header files to included are given below. The purpose of each header file is explained in the following paragraphs.

```
1         #include <stdint.h>
2         #include <stdbool.h>
3         #include "inc/hw_memmap.h"
4         #include "inc/hw_types.h"
5         #include "driverlib/sysctl.h"
6         #include "driverlib/gpio.h"
```

stdint.h: This is used for defining variables according to C99 Standard.
stdbool.h: This is used for defining variables according to C99 Standard.
hw_memmap.h: This contains the marcos defining the memory mapping of Tiva C Series devices. This includes base address locations of all the ports present in Tiva C Series devices. For example, the base address of port B can be accessed by "GPIO_PORTB_BASE".
hw_types.h: This header file includes macros for initialising a pin as input or output. For example, "GPIOPinTypeGPIOOutput" is used for defining a pin as an output pin. For example, "GPIOPinTypeGPIOOutput" is used for defining a pin as an output pin.
sysctl.h : This contains functions and macros for System Control API of DriverLib.
gpio.h: This contains functions and macros for GPIO Control API of DriverLib.

3.2.2 Clock Setup

The TM4C123G contains 2 external clocks. The main oscillator contains a 16 MHz crystal and the Real

Time Clock(RTC) contains a 32.768 MHz crystal. The TM4C123G can be configured to operate at different frequencies. In this case, we will configure the clock to operate at 40 MHz.

```
SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
```

The 16MHz crystal drives the 400MHz PLL. There is default /2 divider in the path. We are further dividing this by 5, which means that the 400MHz PLL is divided by a total factor of 10. Hence the System Clock is set at 40MHz.
Refer to the datasheet to configure the clock at different frequencies.

3.2.3 GPIO Configuration

- **Output**
  The GPIO pins present in TM4C123G can be configured as input, output and interrupt. The clock of TM4C123G has to enabled for a port before using it as an input or output. This can be done by the following line of code.

  ```
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB)
  ```

  In this case, clock is enabled for port B. After enabling the clock, we have to define the status of the pin. Consider an example where pin B6 should be defined as an output pin. This can be done by the following statement.

  ```
  GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_6)
  ```

  Now, let us write a value into the pin B6 which is configured as an output pin. The following lines of code describe the same. In this code logic 1 is written to the pin.

  ```
  GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_6, 0x40)
  ```

- **Input**
  Now we define the same pin(B6) as an input pin. This can be done by the following statement.

  ```
  GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_6)
  ```

  But this not sufficient to use it as an input pin. We have to enable the internal pull-up or pull-down resistor associated with a pin. This is shown below

  ```
  GPIOPadConfigSet(GPIO_PORTB_BASE,GPIO_PIN_6,GPIO_STRENGTH_2MA,GPIO_PIN_TYPE_STD_WPU)
  ```

  The internal pull-down resistor can be enabled by replacing WPU(Weak Pull Up) by WPD(Weak Pull Down).

  The code to read a value from a pin is given below. The value read has to be stored in a variable. In this case, the variable is test.

  ```
  test=(GPIOPinRead(GPIO_PORTE_BASE, GPIO_PIN_6)
  ```

3.2.4 Assignments

- The buzzer is connected to PF4 in Plug and Play board. Write a code to switch the buzzer on and off alternatively.
- The buzzer is connected to PA2 in uC based board. Write a code to switch the buzzer on and off alternatively.
- The motor connections are as shown below. Write a code to move the bot in forward, backward, left and right directions.

Motor Pin	Pin Name(uC)	Pin Name(Plug and Play)
L1	PB0	PF3
L2	PB1	PB3
PWM Left	PF3	PF2
R1	PF4	PC4
R2	PA5	PC6
PWM Right	PA6	PC5